UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

# CS/IT Honours Final Paper 2019


Title: CARPARK – A Low-Cost, Implementable Smart Parking Solution for University Campuses

Author: Joshua Benjamin (BNJJOS003)

Project Abbreviation: CARPARK

Supervisor(s): A/Prof Michelle Kuttel
A/Prof Patrick Marais (2nd Reader)


| **Category** | *Min* | *Max* | Chosen |
|---|---|---|---|
| Requirement Analysis and Design | *0* | *20* | 20 |
| Theoretical Analysis | *0* | *25* | 0 |
| Experiment Design and Execution | *0* | *20* | 0 |
| System Development and Implementation | *0* | *20* | 20 |
| Results, Findings and Conclusion | *10* | *20* | 10 |
| Aim Formulation and Background Work | *10* | *15* | 10 |
| Quality of Paper Writing and Presentation | *10* | | 10 |
| Quality of Deliverables | *10* | | 10 |
| Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*) | *0* | *10* | 0 |
| **Total marks** | | **80** | 80 |

# CARPARK – A Low-Cost, Implementable Smart Parking Solution for University Campuses

Joshua Benjamin
University of Cape Town
BNJJOS003@myuct.ac.za

## ABSTRACT

There is currently a recurring problem on UCT campuses whereby students and staff struggle to find parking daily. In an attempt to provide more information than ever before for people to make better parking decisions, a system has been developed which tracks the availability of parking spots in the parking lots across each campus. The system has been designed with cheap, yet reliable technology in mind, making it exceedingly more costly effective than similar technologies in the field that exist. By doing so, in conjunction with its ease of implementation in any existing infrastructure, it increases its likeliness to be adopted by any system with similar requirements. The testing of the system involved the real-time analysis of a parking lot and the interaction the system had with it. Judging from the tests that have been conducted, the system is only a few modifications away from being completely flawless and accurate as it currently shows promise. As early signs would indicate, the adoption of this system, at its conclusion, will give potential parkers invaluable parking information that they have not had before. The necessity for a system such as this cannot be understated and with CARPARK, there is evidently a revolutionary Smart Parking solution ready to be expanded on and ultimately adopted.

## Keywords

Smart Parking; IoT; Arduino; Software Engineering; Artificial Neural Networks

## 1. INTRODUCTION

The University of Cape Town is one of the biggest universities in South Africa, home to several thousands of students and staff. While there are on-campus residences that students stay in for the duration of the year, there are many who live away from campus and their chosen form of commute to University is to drive. Once on campus, they have to park in certain designated areas that are allocated to students and staff, whereby a parking disc, or permit, is required to park [33]. In order to accommodate both students, of various levels to their degrees and staff, certain parking lots, containing individual parking spots, are specifically available to only those that hold a disc to park in those areas.

As someone who parks on the UCT campus daily, I'm well aware of the frustrations felt by students and staff at the limited amount of parking afforded to us. While several teams in the past have similarly identified the issue at hand, few have suggested an alternative option, within reason, that the UCT Traffic Department would be able to adopt. The issue that currently stands is one where the supply of parking spots is severely limited and relatively constant, while the demand for parking is increasing, as a result of an upward spike in cars on the road, which leads to this problem.

In order to identify potential parking spots, students and staff have had to, in the past, go into their chosen parking lot and survey the area for an available spot. This can often lead to frustration and unnecessary time spent [23], especially when there is no parking found. From both an environmental and economical perspective, there is also an excess amount of fuel wasted in this pursuit for parking [24]. Staff and students are very uninformed in their parking decisions initially and have to take the initiative for themselves to seek this elusive information [14]. It is at this point where the core aims of our project lie, in supplying information to the potential parkers, readily and easily available information.

Thus, I present CARPARK, an in-house system built by two Computer Science Honours students, my project partner Thabo Kopane and myself. The system, in its entirety, is able to determine the availability of parking spots in parking lots where the units are deployed, by tracking the traffic in and out of the parking lots. There is reliable technology behind the physical unit that determines this, with an accompanying app that gives users more insight and information into the available parking. This paper introduces the design rationale of the system and outlines the results of the developed system.

## 2. RELATED WORK

As has been shown through research in the past, potential parkers spend a significant amount of time searching for parking spots in their pursuit for parking, which often leads to frustration [14]. It has also been noted that the provision of parking information to people enables them to make more informed parking decisions themselves, which only adds to their user experience.

Whilst there has been significant work in the area of Smart Parking solutions in the past, a large set of these designs have been for commercial purposes, such as can be commonly found in modern shopping centres [13]. In most of these cases, there are mass infrastructural changes required, which is not a viable option in all environments which would want to be monitored.

A various group of sensors have been used in previous projects such as Infrared sensors [13], Electromagnetic sensors [13] and the more predominantly used, Ultrasonic sensors [9][13]. In some applications, Radio Frequency Identification, or RFIDs, have been used to detect and monitor cars [22]. The cheap, yet reliable, nature of Ultrasonic sensors have led to work being done with them, but as is often the case, they do not allow for a simple solution, but rather focus on a grand solution or failing that, a proof of concept [26].

As is evident by the research conducted, there is room for a solution to be developed that abides by the basic requirements of being low-cost and easily implementable, but none have been done that match what is expected of the proposed system. What is also noticeable is the failure of integrating a mobile app with either a pre-existing system or a new system [16], which is a possibility and therefore is in the scope for this project.

# 3. DESIGN

## 3.1 Aim of Design

With Smart Parking technology available currently, the specific aim of the project and thus, the design of the hardware, was not to research if there is Smart Parking technology. Rather, the research focuses on the possibility of a solution that is both low-cost and easily implementable.

Smart Parking systems, in their current state, require a significant architecture either to exist or to be installed [11], which is considerably expensive. Where this project excels in is the simplicity of the solution and the easy-to-install nature of the technology.

## 3.2 Hardware

After taking into account information gained through research for this project, the Arduino platform, an open-source microcontroller, was chosen as the microcontroller with which the system was to be built. The choice was made on the basis of it being both a cheap but also a reliable technology [2][8], with a wide array of compatible components which would allow for achieving the objectives of the project.

Whilst determining who would be the primary supplier of parts for the project, the price and reliability of suppliers were considered. An initial order was placed with an online store, based in Pretoria, named BotShop. However, parts were needed more on-demand and a distributor, with cheaper parts, in Cape Town was found in Communica. All prices presented below are based on Communica's prices, including VAT, for the parts.

### *3.2.1 Components*

#### *3.2.1.1 Microcontroller*

Initially, the designing and developing of the system began with an Arduino Uno, which is the standard entry level unit for the Arduino microcontroller platform, a device with incredible computing capabilities [29]. Once development began, there were problems with the communication between the Microcontroller and the extendible Wi-Fi module. The issue which was identified was that the Arduino Uno operated at a Transistor-Transistor Logic Level of 5V, while the Wi-Fi module operated at 3.3V. A Logic Level Converter was necessary for this to work. In the process of obtaining the new, required parts, the realisation came that the Arduino Uno, with only a few components to add, was more powerful than the basic operations that was needed from it. Therefore, there had to be an alternative approach to the problem, while identifying which Microcontroller was going to be used.

During research, a microcontroller which combined both the functionality of a microcontroller as well as Wi-Fi capabilities, considering it is a board with a built-in Wi-Fi module, was being presented in internet searches. It provides enough General Purpose Input/Output pins that is needed for the components and the technical specifications are sufficient for the project, with significant specifications being 4MB of memory and a clock speed of 80Mhz [32]. The module, a NodeMCU, was bought for R120 [45], and becomes the primary and most expensive component of the whole system.

Below is an image of the module, in its basic setup, with no added components:


**Figure 1a: NodeMCU Microcontroller**

#### *3.2.1.2 Wi-Fi*

As discussed previously, in order to add Wi-Fi to the project, a component with the capability to connect to an internet network needed to be included. By doing so, it allows the project Internet of Things (IoT) capabilities [2]. The specific component that was used, as it is built into the microcontroller, was an ESP8266-12E. The specifications that are given are also sufficient for the requirements of the project, specifically the relative distance to signal strength. The Wi-Fi standards also subscribe to industry standards with the Wi-Fi type being IEEE 802.11 b/g/n [32]. As tests from hobbyists online have shown, the module maintains 100% connectivity at distances of up to 300 metres [3]. To determine the reliability, this distance was tested and the results showed consistent connectivity at just over 200 metres, which is sufficient for this project. The Wi-Fi module will connect to the nearest Access Point which will be within that range.

#### *3.2.1.3 Sensor*

In order to detect the presence of a vehicle, when it enters and exits a parking lot, a sensor needs to be used. While the research conducted alluded to several sensors that could be used [13], an Ultrasonic sensor was opted for, specifically the HC-SR04 model. This sensor offers both an accuracy and a low-price, ensuring the design aims of the project are met. The sensor contains a trigger point which emits an ultrasound signal, which then reflects off an object for the echo point to receive [21]. The module then determines the distance that the object is at based on the time it takes to receive a response. It can be shown below how the transmission and reception of ultrasonic waves take place:

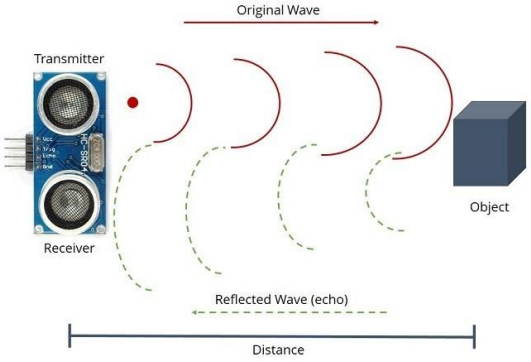



**Figure 1b: Ultrasonic Sensor Operation**

The equation that was used to determine this distance is based on the core principles of the speed of sound as the sensor determines the time it takes to receive a signal in response, in microseconds, to measure the distance. The steps in the algorithm are as follows [21]:

- While we know that the speed of sound, at least in dry air at 20°C is 343 meters per second [28], we need to convert metres into centimetres, considering the distance in centimetres is being measured, which is done by multiplying the quantity with $10^2$, giving centimetres per seconds.
- The next step is to convert seconds into microseconds, which is done by multiplying the quantity by $10^{-6}$, thus giving centimetres per microseconds.
- The final value is 0.0343 centimetres per microseconds which is multiplied with the time it takes to get the distance.
- This value is then multiplied by ½, to compensate for the signal being emitted and received after hitting the object and where only the measure at the reception of the signal is needed, which is double the time it took to hit the object from the transmission. The final equation looks like this:

$$distance = time * \frac{1}{2}(343 * 10^2 * 10^{-6})$$

Only one Ultrasonic sensor is required for each physical unit, at a cost of R33 [44].

### 3.2.1.4 *Logic Level Converter*

As previously mentioned, a logic level converter, or LLC, was used to convert the voltages at which the components communicate [12]. With the reworked system and the new microcontroller, an LLC was needed, in order to communicate between the NodeMCU, operating at 3.3V and the Ultrasonic sensor, operating at 5V. Only one channel of communication was required, with the echo pin, as a 5V logic-level is able to receive communication from a 3.3V, such as with the trigger pin, but not conversely. There is a danger of miscommunication and burning of components if logic levels are not considered closely.

The technology for the LLC works with resistors in series, which are able to convert the voltage of the input either up or down. This works for the input voltage, ground, as well as the communication channels. Below is a diagram of the schematic of the LLC:

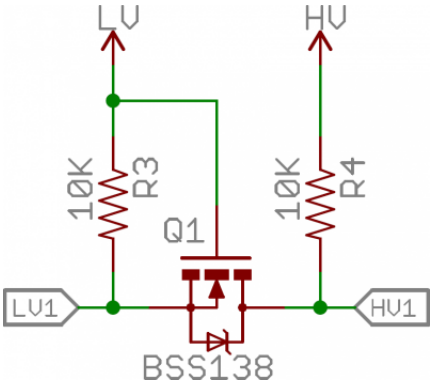



**Figure 1c: Schematic of Logic Level Converter**

A BSS138 model was opted for, which is a 4-channel converter. The specific model was bought in pieces and the pins and board had to be soldered together. This piece cost R20 [42].

### 3.2.1.5 *Breadboard*

In order to compact the size of the physical unit as tightly as possible, the decision was taken to use 2 mini breadboards, each with a set of 10x17 holes. Breadboards are typically used for prototyping purposes but are also capable of handling production environments. Each breadboard cost R15 [43], taking the total for breadboards for each unit to R30.

### 3.2.1.6 *The Complete Unit*

The entire unit assembled cost approximately R200, with a unit having to be deployed at both the entrance and the exit to a parking lot, taking the total cost for physical hardware to around R400 per parking lot.

The figure below, drawn using open-source electronic hardware design tool Fritzing [36], indicates how the whole unit looks with a digital diagram:

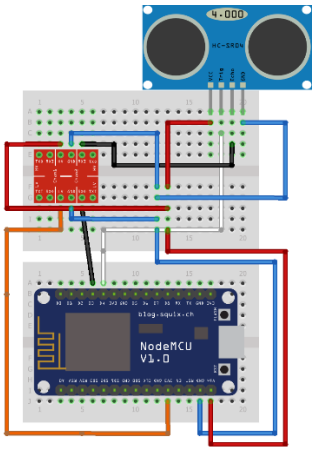



**Figure 1d: Digital Diagram of Unit**

Then, once wired up, the unit looks as below:

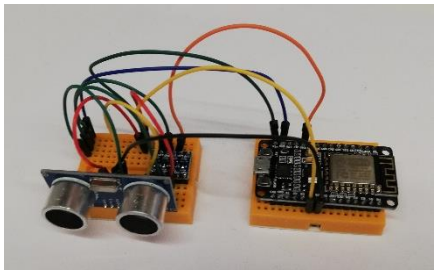


**Figure 1e: Wired Image of Unit**

### *3.2.2 Power Supply*

The unit is powered via the micro-USB slot on the NodeMCU, which then supplies power to the external components. The unit takes in an input voltage of 5V and an input current of up to 2A.

Whilst developing and testing the system, power was supplied via a USB port on a laptop, however in a production environment, there are various ways of supplying power. While testing the system in a production environment, a 15000mAh portable power bank was used that was able to supply the necessary power, for a long period of time. It is proposed that the unit be powered via an AC power supply which is able to supply power constantly with a stable voltage and current.

## 3.3 Software

### *3.3.1 Unit*

Considering the Arduino platform is being used, there is an open-source Arduino Independent Development Environment available [34], which was used in this project. The IDE is used to write code, compile code and deploy compiled code to the Arduino board, in this case, the NodeMCU. The chosen language for this platform is C. While the code was written by myself, additional libraries were used for some components, such as the Wi-Fi library [41].

The single file of code that is required, begins with all the necessary pre-defined constants that will be needed, such as Wi-Fi and Uniform Resource Locator, commonly known as URL, details, defining pins that the components are connected to and a set of weights for an Artificial Neural Network.

The setup function, which is standard as the first function for Arduino boards, is responsible for setting up the network connection with the Wi-Fi module. It uses the network Service Set Identifier, or SSID, and the corresponding password that have been defined before to do so.

The next block of code pertains to the main function loop, which iterates continuously while the board is powered on, with delays between each iteration set at 300 milliseconds. The loop begins with a reading from the Ultrasonic sensor, which is then used to update an array containing the four latest distance readings. With this set of four readings, a detection algorithm method is called. The algorithm chosen was a custom built Artificial Neural Network, with an input layer of four nodes, mapping to the four latest distance readings, a single hidden layer of five nodes and a single output layer, containing one node, with a value ranging between 0 and 1, where the value corresponds to the likeliness of an object being present. Each layer is connected by a set of pre-determined weights. A depiction of the ANN is as follows:

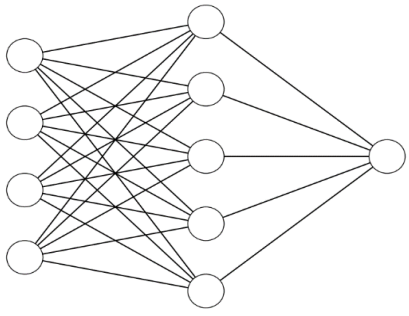


**Figure 2: Artificlal Neural Network**

(Note: This graph was drawn using a slightly altered Python program sourced from GitHub, which takes in the number of nodes per each layer, separated by a comma, as a command-line argument [39])

The reason an Artificial Neural Network was used was to introduce an intelligent way for the system to identify the presence of an object, most notably a car. The nature of this specific ANN allows for easy training of the network with a different set of training data, specific for each implementation and deployment of the physical unit, such as what is required from the parking lot entrance or exit.

Once it has been determined that a car has passed the sensor, whether it be at the entrance or exit of a parking lot, this information is then transmitted to the server via an HTTP POST request, which will be explored more in the following section.

### *3.3.2 Server*

In order to have communication between the hardware side of the project and my partner's application, it was decided that a central server would be used. The purposes of this was the offering of a central point for communication as well as data storage for the activity of the physical unit. It allows a place from where my project partner is able to pull data from.

For the requirements of the project, the Django Framework [35] for web servers was selected as the central server, a framework largely programmed in the Python language. The decision to use Django was based off its open-source and its reliability, being the backend for some of the biggest websites and web servers [30]. The framework also focuses heavily on Object-Orientated Programming principles, which the project team are comfortable with. There is the option of adding a database, using the PostgreSQL query language, which was required. The development environment used for the server was Sublime Text [38], an open-source text editor, while Git [37] was used for version control.

At the creation of the server, a barebone structure is provided, with limited functionality. In order to customise it to the specifications that the project required, an Application Programming Interface (API) had to be developed, to allow for the interaction between the Arduino unit to the database and the database to the mobile application. The API allows for HyperText Transfer Protocol (HTTP) GET requests, to obtain data from the server with a set of parameters [5], and HTTP POST requests, to obtain data from the server with a body of data being sent to the server [5].

#### *3.3.2.1 API Points*

In the next four subsections, all the API points and their functionalities are detailed.

##### *3.3.2.1.1 GET – Parking List*

By accessing this API, data is able to be retrieved about a specific parking lot, such as the number of currently available parking spots in the parking lot. In order to reach this API point, the URL extension to be used is:

*api/Parkings/*

The parameter that the URL takes is a *code* with the parking lot code passed in and so the parameters would be as follows, for example:

- code = P6

This then returns information on the P6, or North Stop, parking lot, such as the capacity and current availability, as well as coordinates of the parking lot.

#### *3.3.2.1.2 POST – Parking List*

In order to update the server with new availability, when a car is encountered, the Arduino reaches this API point, at:

$$api/Parkings/$$

The data that is passed through the body of this HTTP POST request is in the form of x-www-form-urlencoded, with parameters of *code*, pertaining to the parking lot which it applies to and *status*, which is either "ENTER" or "EXIT", based on the activity of the car.

An example of a completed POST request would be to the API extension above, with the body values of:

- code = P6
- status = ENTER

The result of this would update the server, decrementing the number of parking spots available in the P6 parking lot as a car has just entered. Additionally, an HTTP code is returned from the server, with a code of 200 [5] indicating that the request was made successfully and the server has processed it.

#### *3.3.2.1.3 GET – Parking Lot List*

In order to display the information on parking lots, as seen in 3.*3.2.1.1 GET – Parking List*, but for an entire campus, this API is called at the extension:

$$api/Parking - Lots/$$

The parameter that is passed through in this GET request is the name of the campus, for which data is required. An example of the parameter would be as follows:

- campus = Upper Campus

The result of this request would return information on all the parking lots that are on Upper Campus.

#### *3.3.2.1.4 GET – Parking Days*

A future development for the app is conducting statistical analyses of how people park, based on historical data. This API point allows for the retrieval of parking data from the day before the request is made, unless it is Sunday or Monday, for then, data collected from the previous Friday is used. The API extension is as follows:

$$api/Parking - \text{Days}/$$

The two parameters that are passed in are the current day as well as the code for the parking lot for which the data is requested. These parameters look as follows:

- code = P6
- day = Thursday

The result of this request is a list of all the parking that occurred in the P6 parking lot on the day before Thursday, which would be Wednesday. Statistical data can then be collected to display the times on the app when the parking lot is at capacity.

#### *3.3.2.2 Deployment of Server*

To further enhance the possibility of having this new system be implemented at the culmination of the project, the server was deployed to a real-time hosting website, Heroku. While there are certain restrictions to the free account that was used, such as a limit of 20 connections at a time and a timeout of the website after inactivity of more than 30 minutes, it works in the context that is required for testing purposes for this project [31]. There is an option to upgrade the account which allows for more capabilities, if the system is to be scaled and include more parking lots. The current web address that the server is hosted at is:

$$uctcarpark.herokuapp.com$$

# 4. RESULTS AND DISCUSSION

In this section of the paper, the results of the experimentation and implementation of the system are addressed, and results are formulated. Following the results will be discussions concerning the results, indicating the successes and failures.

## 4.1 Design

At the outset of the project, one of the main considerations was that technology which track parking in large developments, such as shopping centres, are an expensive technology. As can be seen in the section detailing the physical design of the module, the cost of this technology has been limited to a cost of R400, for two units. Each parking lot on the system will require two units, for both the entrance and exit. This is in stark contrast to industry standard Smart Parking solutions, which despite the considerable price, still require a vast infrastructure overhaul.

From a purely design perspective, the unit has been designed in a way that all the requirements that were set out for it were met, it is both low-cost and reliable. While the design may be improved over time, and parts made cheaper, the unit is at a satisfactory stage now with its first iteration.

## 4.2 Experimentation and Implementation

In order to reliably determine the future of this project, experimentation needed to be done on the system in its entirety to gauge the successes and failures. The testing phase was done in the three iterations detailed below.

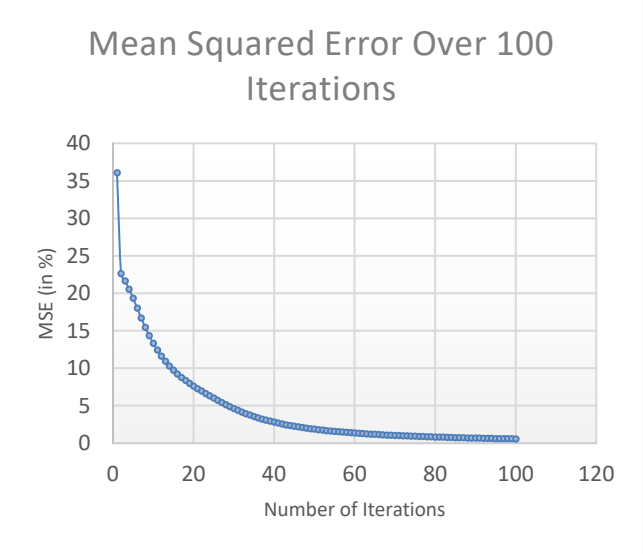
### *4.2.1 Iteration 1 – Requirements Gathering*

For the first iteration of tests, the unit was built as it appears in its final state, with very minimal code. The purpose of this iteration was to gather testing and training data for the Neural Network. The unit was placed at the entrance to a parking lot and documented how the distance measurement took place when a car drove past. During this experimentation, when there was no object in the way of the sensor, the system would output that there was an object at a distance of between 2200 centimetres and 2300 centimetres away, which is far the sensors 400 centimetres range. These distances were therefore able to be nullified. In the event of a car driving past, the distances measured were between 50 centimetres and 110 centimetres, thus giving a wide enough distance range for testing data.

#### 4.2.1.1 *Error in Training Neural Network*

For the training of the Neural Network, a Python program sourced online [40] was used, with slight modifications to satisfy the requirements of this project. A set of custom training inputs and outputs was used that represented the real-life data from the interaction with cars. These values were obtained by observing the distances that the sensors picked cars up at as they drove past, whilst having to physically determine whether there was a car present.

With this set of training data, the network was trained using feed-forward methods, where inputs are multiplied with their respective weights to obtain a new value for each neuron at the next layer [10]. This value at each neuron is then passed forward another iteration to the output layer. At the output layer, if the Mean Squared Error (MSE), which is the difference between the expected output and the given output, is at a small enough value chosen, the Neural Network is considered trained. However, in the case where there is a greater error, we use a technique referred to as backpropagation, which takes the MSE and adjusts the weights leading to that neuron to better fit the expected output. This is accomplished by using gradient descent [1], a mathematical model which finds the local minima of a function, over a number of iterations. The training algorithm that was used was run for a finite number of iterations, but it could also have been run until the MSE was deemed small enough. Below is a visual representation of the nature of the redefining of the weights in order to minimise the MSE:




**Line Graph 1: Mean Squared Error Over Iterations**

While the training algorithm was run for 1000 iterations, with an eventual Mean Squared Error of 0.03%, after 100 iterations, this graph displays a Mean Squared Error of 0.6%.

### 4.2.2 *Iteration 2 – Detection Algorithm Validation*

The next phase was with a fully built unit too, however the code was altered to encompass the detection algorithm. The code for this iteration is simply just the feed-forward segment of the Neural Network. The four latest distances that the Ultrasonic sensor measures are used, after each measurement, approximately every 300 milliseconds, and in conjunction with the pre-determined weights, the algorithm yields an output value ranging from 0 to 1, to determine whether there is a vehicle present or not.

At the first observation, 100% of the vehicles that passed the sensor were accurately detected as passing the sensor, however an issue that is expanded on further in this paper was also identified, whereby at times, cars driving on the other side of the road were also picked up by the sensor. In order to minimise this, the training set for the detection algorithm was adjusted to limit the maximum distance the sensor would recognise a car at.

### 4.2.3 *Iteration 3 – The Complete System*

For the third and final iteration, the system in its current state was tested at a parking lot on campus for a full hour, to replicate an hour of activity at that parking lot. This system contained code that allowed for the adjusted detection algorithm, in conjunction with network capabilities, to determine when a car interacts with the system, by way of entrance and exit, to be tracked and uploaded to the server. At each detection of a car, the time it took to upload the data to the server was between 300 and 600 milliseconds, which is impressive considering it takes at least 1500 milliseconds for one car to enter or exit after another. This is a success as it means that there will be no foreseeable network bottleneck whereby uploads will hang and wait for previous uploads to complete.

#### 4.2.3.1 *Detection Algorithm Correctness*

In order to judge the correctness of the detection algorithm, data was obtained from cars that drove past each unit and the results can be seen below:

| | **True Positive** | **False Positive** | **False Negative** | **Total** |
|---|---|---|---|---|
| **ENTER** | 35 | 4 | 2 | **41** |
| **EXIT** | 24 | 4 | 9 | **37** |

**Table 1: Results From System Interaction**

The distinction between **True Positive, False Positive** and **False Negative** need to be made for an understanding of these results. A **True Positive** indicates that a car drove past the unit and it was picked up by the system, having its data uploaded to the server. A **False Positive** would be if the system identified that a car drove past the sensor, where it did not. A **False Negative** occurs when a car drives past the sensor, but the system fails to identify this.

(Note: there is no need for **True Negatives** as the system currently does not operate to actively seek negatives and therefore, they have no place in the data)

What is immediately apparent is that the system did not accurately detect 100% of car traffic, both at the entrance and exit of the parking lot. Interpreting the **ENTER** data, it identified 35 out of 41 True Positives, an 85% success rate. There were however four False Positives, as a result of people walking past the sensors and two False Negatives, whereby the car was not picked up as entering the parking lot. This is encouraging work considering the high accuracy of the True Positives, however the other values need to be limited as much as possible. The False Negatives would be limited by attending to the detection algorithm, notably the training set, to ensure that it is capable of detecting cars every

time they drive past the sensor. In order to limit the False Positives, the system would need to take into account when a human or alternative object moves past the sensor and ignore this activity. More detail on how this would be accomplished is found in the next section of this paper.

The results for the **EXIT** data are seen to be worse, with only a 65% success rate in identifying cars driving past the sensor, 24 out of 37. There are the same number of False Positives, as a result of people walking past the sensor as well, but there are more False Negatives, with nine times where a car drove past the sensor undetected. The correction of the False Positives would be the same as the above, for ENTER, however corrections need to be made to significantly decrease the number of False Negatives. In the detection algorithm for the EXIT unit, the same training set, and thus same weights, were used as with the ENTER unit. The assumption was such that both units would detect cars at certain distances exactly the same, however these results indicate that this is not true. In order to correct this, a new training set would have to be produced for the exit unit, with values that pertain to the exiting behaviour of cars.

# 5. RESTRICTIONS TO SUCCESS

## 5.1 Safety of Module

The first of the fears as to whether this can be a fully implementable system is with regard to the safety of the physical module. This comes in two forms, the safety from the weather and the safety from theft.

### *5.1.1 Environmental*

In order to protect the unit from any environmental damage, a special case will have to be designed that is able to prevent any damage. Such a case should be able to both contain all of the components as well as prevent anything from getting inside, except for through customised holes for access to the Micro-USB slot and the 2 nodes on the Ultrasonic sensor.

### *5.1.2 Theft*

Another issue to consider is the safety of the module, in terms of the protection and prevention of theft. With a rise in the last year of car theft on UCT campuses, introducing a technology which could be easily stolen might be problematic.

There are two methods where issues could be prevented. The first would be to design the holding case in a way that it camouflages into the position it is strategically placed in. By doing this, it would not be an instantly noticeable unit, which might help in preventing theft.

Another suggestion would be to build the unit either into a physical holding that would not be able to be broken into and stolen or a less obtrusive method of building into the ground, still ensuring that the sensors are able to pick up cars.

## 5.2 Obstructions to the Unit

A difficult issue to resolve is the unit picking up on an object that is placed in front of the sensor. In this case, it will only register as a car entering/exiting the parking lot, once it is no longer in front of the sensor.

This brings up a few potential scenarios that would need to be prevented. In the event of a human blocking the ultrasonic sensor, a passive infrared sensor could be installed to prevent the unit from registering it as a car passing through [27]. The way the infrared technology operates, specifically for this component, is it constantly emits an infrared beam and if there is contact in its range that emits body heat, such as a human or animal, it registers to the infrared sensor. The system would then be able to avoid picking up on the human body this way.

The second scenario would be if an object is placed in front of the sensor, while cars pass behind it. The system would wait for the object to move or be moved away before triggering an update to the count, while all the cars entering or exiting the parking lot would go untracked. To prevent this from occurring, a check in the detection algorithm would have to be introduced whereby it notices a continuously static object in front of the sensor.

## 5.3 Driving Behaviours of People

A key problematic area when considering the impact of experiments containing humans as the subjects, is how they will interact with the system. The inability of the person to act as required by the system is not a fault of the person but rather the fault of the system.

During a round of testing of the implementation in a car parking lot, an unexpected behaviour was observed of some of the drivers. While entering the parking lot, considering there are no physical barriers at the entry and exit points, only indications by way of painted lines, drivers were entering or exiting on the wrong side of the road. Despite the legality of this, it also deceived the system as a car entering the parking lot by way of the exit would not only fail to register as an entrance but would also trigger the exit sensor to push that data to the server. This disrupted the accurate counting of car traffic twice as badly. The same is true for the opposite action, a car exiting on the wrong side of the road.

In order to prevent this, there are minor adjustments to be made to the software. While altering the detection algorithm to only allow for cars to be recognised at a much shorter distance would help, it does not solve the issue. This improvement would help whereby a car does not fully enter in the correct lane and is partially over the lane divider, however the problem is not wholly preventable and persists when the someone drives fully in the opposite lane.

There is a hardware modification that could be made that would enhance the accuracy. By placing an additional Ultrasonic sensor at a different angle, a forward-facing angle, we would be able to determine whether a car is moving towards the sensor or not, which would indicate their direction.

A more infrastructurally enforced change would be to build a physical barrier in between the two lanes, such as a pole or a row of poles, thus limiting the driver's aversion to the road rules and ensures they fully enter or exit in one of the two lanes, most likely the one intended for their action.

## 5.4 Improper Parking

An issue that has been identified is with regards to how people park in certain parking lots. The system assumes that the parker will abide by the rules set out by the parking authority and does not consider that they may break these rules. A way of breaking the rule, as an example, would be double parking, which means occupying two parking spaces with just one car. The system will assume that there is a space available because only one car entered, despite it occupying two spots. There is nothing that can be done to prevent this with the current system.

The second issue, which is prevalent at UCT, is when students park in a parking lot in an open space that they deem good enough to be a parking spot, which it is not. The downside to this is that the system will, overtime, lose its accuracy as the number of cars inside the parking lot will not represent the true availability because of these certain cars.

While the prevention of these acts is limited at this stage, currently, UCT Traffic are responsible for awarding fines to people who park improperly. The problem seems to be that it is not taken seriously enough and does not happen frequently enough to be reliable. If UCT Traffic were to alter their approach, possibly with the use of something like this system, which notifies them when there is an excess number of cars in a parking lot for an extended period of time, then people would cease to park improperly.

## 5.5 Alternative Parking Spots

One of the biggest concerns going into the project was having to consider the alternative types of parking spots available in parking lots. A perfect example of this is disabled and/or wheelchair parking spots. Because these are present in parking lots, they afford people who need the spots to park there, however the parking is only for specific people, who use the exact same entrance and exit as everyone else. Taking a case scenario of someone, who is permitted to park in a disabled parking, entering the parking lot. The system would be triggered by this action, and would decrement the number of parking spots available, despite the person not occupying a parking spot that is freely available, making the count inaccurate.

While there is no fix in the current state that the project is in, this could be achieved by two of the methods outlined below. The first being if there were additional sensors placed by all disabled parking spots, which would be able to determine whether a car is in that space or not and then make the necessary adjustments to the count on the parking spots, on the server side. The second and less pervasive suggestion, would be to encourage the usage of the CARPARK mobile app, where there would be a functionality for people registered to park in disabled parking bays to indicate that they have parked there. The app could then inform the server via an API point and then the necessary updates would be made.

# 6. CONCLUSIONS

At the outset of this project, we were determined to gain more information and understanding into a problem that is so prevalent and unnecessary for most people who park on campus daily, as explained in the introduction. While we knew there were going to be challenges, considering that if there were not, there would have been a system in place already, but we decided to tackle the problems head on in order to build a solution.

In terms of the design of the whole system, it achieved the aims that was set out for it, in terms of it being a low-cost and reliable technology. In addition, the sheer simplicity of the system and with the right system architecture put in place already, this solution is scalable to allow for the system to include as many parking lots as necessary. There are financial limitations, as each parking lot costs an additional amount to be spent on the units and purchasing more server space is required, but this would be up to the discretion of the implementer of the system.

With regards to the code behind the system, there are still improvements to made to ensure 100% accuracy and to eliminate any false positives or false negatives that the system identifies. The brunt of this will come in the form of reworking the detection algorithm, most notably a more considered and specific training set for each parking lot and entry or exit point.

At this stage, the system is not ready to be deployed, as was hoped for when the project began, but it is only a couple of iterations away from being a fully functioning and reliable system. The restrictions to success do however need to seriously be taken into consideration with regards to error correcting if necessary.

This project has been a success in the sense that it has laid down some solid ground work, with clear direction for the future of the project. Hopefully in time, the solution can be expanded on, providing parkers on UCT campuses with a more informed and more enjoyable parking experience.

# 7. FUTURE WORK

Throughout this paper, improvements have been highlighted which will have to be made should this become a fully implementable system. In this following section, I discuss the future of the overall system which would add an extra dimension to its functionality.

## 7.1 Display Board

At the proposal of this project, the goal was to include an LED display board in conjunction with the Arduino system. On this display would be the availability of parking at the parking lot that it is stationed at. This would, together with the app, give all users full information over parking lots, allowing them to make an even more educated parking choice than ever before.

The primary reason that this did not go ahead was the price of the display board. The goal was to build it in-house, however I did not have the expertise and the parts that were required turned out to be too expensive for the low-cost oriented project.

A prototype has been developed with the use of a smaller LCD screen that is able to retrieve data from the server, displaying the available parking. This serves as a proof of concept that this is possible, at a larger scale. The wiring of this system looks as shown below:

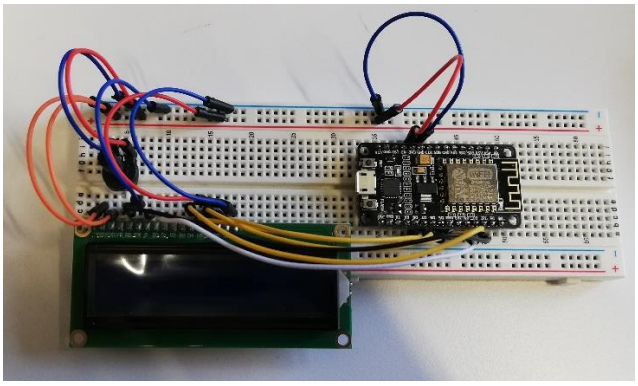


**Figure 3: LCD Reading Data from Unit**

## 7.2 Unit Case

While the testing of the system has been with prototyped units, once there is a final unit that will be deployable, having the wires open and all components freely available will not suffice. For reasons stated in the section pertaining to the security of the unit, there needs to be a case, protecting the unit and all of its components.

This case will have to be designed and produced with all considerations in mind, such as cost, protection and visible aesthetic. The unit will then be placed in the case, with slots available for the necessary components such as power and ultrasonic trigger and echo points.

## 7.3 Number Plate Recognition

While the work that has been done on this project pertains to incredibly basic parking behaviour, obtaining simple information, there are future applications that can be implemented that are more complex and utilise some existing and powerful technologies.

An example of one of these would be Image Recognition, given the advancements in this specific field. While conducting research I came across a relatively efficient method of processing an image of a car, including its number plate, and then training a Convolutional Neural Network [7][15] to identify the number plate in the picture. This would then be combined with Optical Character Recognition technologies in order to retrieve the number plate of a car entering and exiting the parking lot in raw text [20][25].

Introducing this technology would have many applications, one that has been hypothesised is a reworking of the parking payment plan [25]. In its current state, students and staff are required to purchase parking discs to park [33], however, if we could obtain information on individuals, a revolutionary pay-as-you-park system could be introduced. This would function where the amount owed, for time spent parking, at the end of the month, is charged to a linked UCT account.

This is an extension that is not easily and quickly possible to integrate, however there is a future for it if the rest of the CARPARK system is adopted and widely accepted by the greater UCT community.

## ACKNOWLEDGEMENTS

I begin my acknowledgements with thanks to my project partner, Thabo Kopane. Together we were able to push each other throughout the project and provide help where we both needed it. It was a pleasure to work with you.

In addition, I would like to express gratitude to my supervisor, Professor Michelle Kuttel, for her constant guidance and support throughout the lifetime of the project and for imparting invaluable knowledge. As well, I would like to thank Professor Patrick Marais for his role as second reader and for the advice and guidance in which he gave during the project.

Lastly, I would like to thank the Computer Science Department at UCT, specifically Craig Balfour, for allowing me to use their equipment to solder the necessary parts together.

## REFERENCES

*Research*

[1] Andrychowicz, M. 2016. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, 29.

[2] Barbon, G. and Margolis, M. 2016. Taking Arduino to the Internet of Things: The ASIP programming model. *Computer Communications*, 89-90, 128-140.

[3] Benchoff, B. 2014. *ESP8266 Distance Testing*. Hackaday. Available at: https://hackaday.com/2014/09/26/esp8266-distance-testing/

[4] Benson J.P., O'Donovan T., et al. 2006. Car-park management using wireless sensor networks. *Proceedings of 31st IEEE Conference on the Local Computer Networks*, 588–595.

[5] Berners-Lee, T. 1999. Hypertext Transfer Protocol -- HTTP/1.1. *RFC 2616*.

[6] Choeychuen K. 2013. Automatic parking lot mapping for available parking space detection. *Proceedings of the 5th International Conference on Knowledge and Smart Technology (KST)*, 117–121.

[7] Delmar Kurpiel, F., Minetto, R. and Nassu, B. 2017. Convolutional neural networks for license plate detection in images. *2017 IEEE International Conference on Image Processing (ICIP)*.

[8] Ferreira, H. and Dias Canedo, E. 2013. IoT architecture to enable intercommunication through REST API and UPnP using IP, ZigBee and arduino. *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*.

[9] Grodi, R., Rawat, D. and Rios-Gutierrez, F. 2016. Smart parking: Parking occupancy monitoring and visualization system for smart cities. *SoutheastCon 2016*.

[10] Gurney, K. 1997. *An Introduction to Neural Networks*. 1st edition. Taylor & Francis, London.

[11] Ji, Z., Ganchev, I., O'Droma, M., Zhao, L. and Zhang, X. 2014. A Cloud-Based Car Parking Middleware for IoT Based Smart Cities: Design and Implementation. *Sensors*, 14(12), 22372-22393.

[12] Jimblom. 2013. *Bi-Directional Logic Level Converter Hookup Guide*. Learn.sparkfun.com. Available at: https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide/all.

[13] Khanna, A. and Anand, R. 2016. IoT based smart parking system. *2016 International Conference on Internet of Things and Applications (IOTA)*, 266-270.

[14] Khattak, A. and Polak, J. 1993. Effect of parking information on travelers' knowledge and behavior. *Transportation*, 20(4), 373-393.

[15] Kim, H. S., et al. 1991. Recognition of a Car Number Plate by a Neural Network. *Proceedings of the Korea Information Science Society Fall Conference*, 18, 259-262.

[16] Kopecky, J. and Domingue, J. 2012. ParkJam: crowdsourcing parking availability information with linked data. *The Semantic Web: ESWC 2012 Satellite Events*, 381-386.

[17] Krogh, A. (2008). What are artificial neural networks?. *Nature Biotechnology*, 26(2), 195-197.

[18] Lee, S., Jo, J., Kim, Y. and Stephen, H. 2014. A Framework for Environmental Monitoring with Arduino-Based Sensors Using Restful Web Service. *2014 IEEE International Conference on Services Computing*.

[19] Marquez, M., Lara, R. and Gordillo, R. 2014. A new prototype of smart parking using wireless sensor networks. *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*.

[20] Mori, S., Nishida, H. and Yamada, H. 1999. *Optical Character Recognition*. 1st edition. Wiley-Interscience, New York.

[21] Narkar, A. 2019. *Simple math behind calculating distance using Ultrasonic sensor HC-SR04*. Medium. Available at: https://medium.com/@adityavijaynarkar/simple-math-behind-calculating-distance-using-ultrasonic-sensor-hc-sr04-66ed5a6aa214.

[22] Pala, Z. and Inanc, N. 2007. Smart Parking Applications Using RFID Technology. *2007 1st Annual RFID Eurasia*, 1-3.

[23] Shoup, D. 1997. The High Cost of Free Parking. *Journal of Planning Education and Research*, 17(1), 3-20.

[24] Shoup, D. 2006. Cruising for parking. *Transport Policy*, 13(6), 479-486.

[25] Sirithinaphong, T. and Chamnongthai, K. 1999. The recognition of car license plate for automatic parking system. *ISSPA '99. Proceedings of the Fifth International Symposium on Signal Processing and its Applications (IEEE Cat. No.99EX359)*.

[26] Srikanth, S. and Patil, M. 2009. Design and Implementation of a Prototype Smart PARKing (SPARK) System Using Wireless Sensor Networks. *2009 International Conference on Advanced Information Networking and Applications Workshops*, 401-406.

[27] Tomooka, H. and Sugimoto, T. (2019). *US5703368A - Passive-type infrared sensor system for detecting human body - Google Patents*. Patents.google.com. Available at: https://patents.google.com/patent/US5703368A/en

[28] Wong, G. 1986. Speed of sound in standard air. *The Journal of the Acoustical Society of America*, 79(5), 1359-1366.

*Information*

[29] Arduino Uno. Available at: https://components101.com/microcontrollers/arduino-uno

[30] Django Websites. Available at: https://codecondo.com/popular-websites-django/

[31] Heroku Platform. Free account specifications. Available at: https://www.heroku.com/pricing

[32] NodeMCU. Available at: https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/

[33] UCT Parking. Available at: http://www.students.uct.ac.za/students/services/transport-parking/parking

*Software*

[34] Arduino IDE. Version 1.8.9. Available at: https://www.arduino.cc/en/main/software

[35] Django. Version 2.2.4. Available at: https://www.djangoproject.com/download/

[36] Fritzing. Version 0.9.3b. Available at: https://fritzing.org/home/

[37] Git. Version 2.23.0. Available at: https://git-scm.com/downloads

[38] Sublime Text. Version 3.2.1. Available at: https://www.sublimetext.com/3

*Code*

[39] ANN Drawing in Python. Available at: https://gist.github.com/craffel/2d727968c3aaebd10359#file-draw_neural_net-py

[40] ANN Training in Python. Available at: https://enlight.nyc/projects/neural-network/

[41] ESP8266 Library. Available at: https://github.com/esp8266/Arduino

*Components*

[42] 4-Channel Logic Level Converter. https://www.communica.co.za/products/bsk-logic-level-convertor-new

[43] Breadboard. https://www.communica.co.za/products/bsk-mini-breadboard-yellow

[44] HC-SR04 Ultrasonic Sensor. https://www.communica.co.za/products/bsk-ultrasonic-sensor-hc-sr04

[45] NodeMCU. https://www.communica.co.za/products/acm-esp8266-nodemcu-wifi-board